



Algorithm Theory

Sample Solution Exercise Sheet 11

Due: Friday, 23rd of January 2025, 10:00 am

Exercise 1: Balls into Bins

(10 Points)

Assume we have n bins and n balls (for $n \geq 2$). We now throw all the balls uniformly at random into the bins. In the following we want to show that the maximum number of balls per bin is at most $O(\log n)$ with high probability. For that we define the *maximum load* L by $\max_{1 \leq j \leq n} Y_j$ where (random variable) Y_j stands for the number of balls in bin j .

(a) For a given bin j , what is the expected number of balls in j ? (i.e., compute $E[Y_j]$) *(2 Points)*

(b) Use a Chernoff Bound to show that $P(Y_j \geq 2e \cdot \log_2 n) \leq 1/n^{2e}$. *(6 Points)*

Chernoff Bound: Suppose X_1, X_2, \dots, X_N are independent random variables taking values in $\{0, 1\}$. Let X denote $\sum_{i=1}^N X_i$ and let $\mu = E[X]$ be this sums expected value. Then for any $\delta > 0$,

$$P(X \geq (1 + \delta) \cdot \mu) \leq \left(\frac{e^\delta}{(1 + \delta)^{(1+\delta)}} \right)^\mu$$

(c) Show that the maximum load L is small, i.e., show that $P(L < 2e \cdot \log_2 n) > 1 - \frac{1}{n^4}$. Use a Union Bound! *(2 Points)*

Sample Solution

a) Let X_i for $1 \leq i \leq n$ be the random variable that is 1 if ball i was thrown in bin j and else is 0. By our assumptions we have that for a given j all X_i are independent and $P(X_i = 1) = 1/n$. Thus,

$$E[Y_j] = E \left[\sum_{i=1}^n X_i \right] = \sum_{i=1}^n E[X_i] = \sum_{i=1}^n (0 \cdot P(X_i = 0) + 1 \cdot P(X_i = 1)) = \sum_{i=1}^n \frac{1}{n} = 1$$

b) By the previous task we know that $\mu = E[Y_j] = 1$. Also note that by the assumption that $n \geq 2$ we have that $\log_2 n \geq 1$. We can now proof the statement by choosing $\delta = 2e \cdot \log_2 n - 1$:

$$\begin{aligned} P(Y_j \geq 2e \cdot \log_2 n) &= P(Y_j \geq (1 + \underbrace{2e \cdot \log_2 n - 1}_{\delta}) \cdot \mu) \\ &\leq \left(\frac{e^{-1} \cdot e^{2e \cdot \log_2 n}}{(2e \cdot \log_2 n)^{2e \cdot \log_2 n}} \right)^1 \\ &= \frac{1}{e} \cdot \left(\frac{e}{2e \cdot \log_2 n} \right)^{2e \cdot \log_2 n} \\ &\leq \left(\frac{1}{2} \right)^{2e \cdot \log_2 n} = \frac{1}{2^{\log_2(n^{2e})}} = \frac{1}{n^{2e}} \end{aligned}$$

c) If there is some j s.t. $Y_j \geq 2e \cdot \log_2 n$, the maximum load conflicts the statement of the task. We will use a union bound to estimate the probability that there exists such a j :

$$P(L \geq 2e \cdot \log_2 n) = P \left(\bigvee_{j=1}^n (Y_j \geq 2e \cdot \log_2 n) \right) \leq \sum_{j=1}^n P(Y_j \geq 2e \cdot \log_2 n) \leq n \cdot 1/n^{2e} = 1/n^{2e-1} \leq 1/n^4$$

Exercise 2: All Zero

(5 Points)

You are given an array A of length n filled with bits $\{0, 1\}$. We want to know if all entries in A are 0, or if "sufficiently many" are 1.

More concrete, we want an algorithm that returns **all-zero** if all entries of A are 0, and that returns **many-ones** if at least $\lceil \sqrt{n} \rceil$ entries are 1. In the remaining cases (i.e., the number of 1's is between 1 and $\lceil \sqrt{n} \rceil - 1$) you can return either of the two answers.

The goal is to find a randomized algorithm that solves the above problem in sublinear time and with success probability at least $1 - 1/n$. Argue correctness and prove the success probability.

Note: You will not get points for linear algorithms. The idea is to subsample k entries of A and compute an output based on only these sample points. Try to choose k as small as possible. Furthermore, you may want to use inequalities stated in the [lecture slides](#).

Sample Solution

Algorithm: We pick k indices from A independently and uniformly at random. If all of them are 0 we return **all-zero** otherwise we return **many-ones**.

Correctness: Note that the algorithm is always correct if A contains only 0's (as the sample will also contain only 0's) or at most $\lceil \sqrt{n} \rceil - 1$ many 1's (as we are allowed to return any result). However, if A contains $\geq \sqrt{n}$ many 1's the algorithm can fail. We will now bound the error probability in this case. So let A contain $\geq \sqrt{n}$ many 1's we have to bound the probability that our sample consists of 0's only. Let p be the probability that a sample point is 1. In this case we have $p \geq \frac{\sqrt{n}}{n} = \frac{1}{\sqrt{n}}$. We will now show that the failing probability is $\leq 1/n$ if we set $k := \lceil \sqrt{n} \cdot \ln(n) \rceil$.

$$P(\text{fail}) = (1 - p)^k \leq \left(1 - \frac{1}{\sqrt{n}}\right)^{\sqrt{n} \cdot \frac{k}{\sqrt{n}}} \leq e^{-1 \cdot \frac{k}{\sqrt{n}}} \leq e^{-\ln(n)} = \frac{1}{n}$$

It remains to argue about the runtime. Since we have to check all k sampled entries, the runtime is $O(k) = O(\sqrt{n} \log(n))$ and hence clearly sublinear.

Exercise 3: Set of Strangers

(5 Points)

Consider a group of n people. Suppose there are $2n$ pairs of people of this group that know each other. We may model this situation as an undirected graph $G = (V, E)$, in which the vertices are people, and we draw an edge between two people if and only if they know each other. The total number of edges is therefore $2n$. We want to find a huge group of strangers, i.e., a set of nodes where no edges connect nodes within this set. To achieve this we use the following randomized algorithm:

- First, every node joins a set $S \subseteq V$ independently with probability $1/4$.
- Let $E_S \subseteq E$ be the set of edges where both endpoints are in S .
- Now, iterating through the edges $e \in E_S$ and deleting one arbitrary endpoint of e from S (if not already deleted) gives us the final set of strangers S' .

Show that $E[|S'|] \geq n/8$.

Hint: The number of people getting deleted from S can not be more than $|E_S|$.

Sample Solution

It is clear by the construction that $E[|S|] = n/4$. For an edge $e = (u, v)$, the probability that $e \in X_S$ is, due to independence, the product of the probabilities that $u \in S$ and $v \in S$, i.e., $P(e \in E_S) = 1/16$. Summing over all $2n$ edges gives $E[|E_S|] = 2n \cdot 1/16 = n/8$. Using the statement of the hint, we can end the proof as follows $E[|S'|] \geq E[|S|] - E[|E_S|] = n/8$.